Summarizing Data: Relationships Between Variables in R

In this document we show how to find relationship between two variables. Since quantitative and categorical variables must be treated differently we show which plots, tables and statistics are appropriate in different cases.

Relationship between quantitative and categorical variables

We start this section with 'Life Expectancy' data set. First we download these data into R using 'read.table' function (since there is a header in this file we must put 'header=TRUE' in the argument). Next we print the first 6 observations using 'head' function and then use 'attach' to use each variable individually:

```
LifeExp.data = read.table('LifeExpComplete.txt',header=TRUE)
head(LifeExp.data)
attach(LifeExp.data)
```

```
      Country Region LifeExp       GDP HIV
1 Afghanistan    SAs  48.673        NA  NA
2     Albania   EuCA  76.918        NA  NA
3     Algeria   MENA  73.131  6406.817 0.1
4      Angola    SSA  51.093  5519.183 2.0
5   Argentina   Amer  75.901 15741.046 0.5
6     Armenia   EuCA  74.241  4748.929 0.1
```

We want to understand the relationship between 'Life Expectancy' and 'Region'. First we find basic statistics of 'LifeExp' for East Asia and Pacific region ('EAP'). We use familiar 'summary' function:

```
summary(LifeExp[Region=='EAP'])
```

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  62.48   68.77   72.95   73.09   77.63   83.39
```

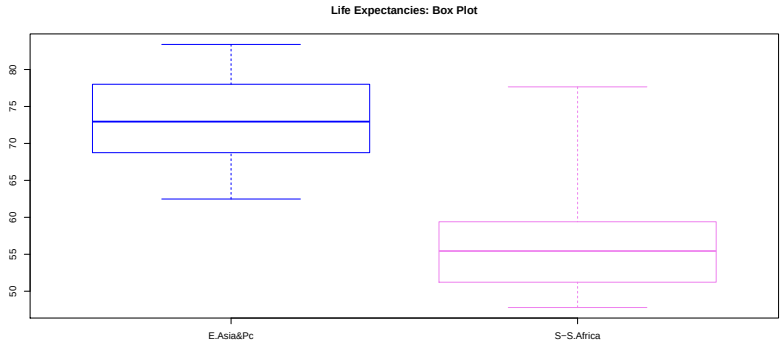
Here 'LifeExp[Region=='EAP']' means that we take into account only those values of life expectancy for which region is 'EAP'. Next we get exactly the same statistics but for the 'SSA' region:

```
summary(LifeExp[Region=='SSA'])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.79   51.22   55.44   56.80   59.40   77.65
```

Finally we make boxplots of 'Life Expectancy' for each of these two regions ('list' function attaches two data sets together, 'range=0' means that we construct a simple boxplot, 'names' gives names to each boxplot on the horizontal axis and 'border' function gives color of boxplots' borders):

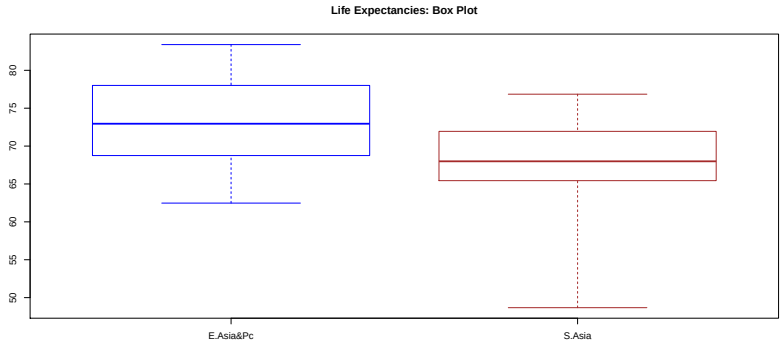
```
boxplot(list(LifeExp[Region=='EAP'],LifeExp[Region=='SSA']),range=0,
  main='Life Expectancies: Box Plot',names=c('E.Asia&Pc','S-S.Africa'),
  border=c('blue','violet'))
```



Similarly we find basic statistics for 'SAs' region and compare it with 'EAP' using boxplots:

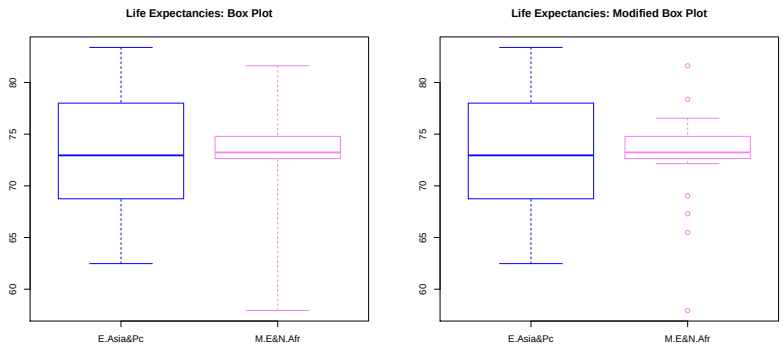
```
summary(LifeExp[Region=='SAs'])
boxplot(list(LifeExp[Region=='EAP'],LifeExp[Region=='SAs']),range=0,
  main='Life Expectancies: Box Plot',names=c('E.Asia&Pc','S.Asia'),
  border=c('blue','brown'))
```

```
summary(LifeExp[Region=='SAs'])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  48.67   65.44   67.99   67.03   70.44   76.85
```



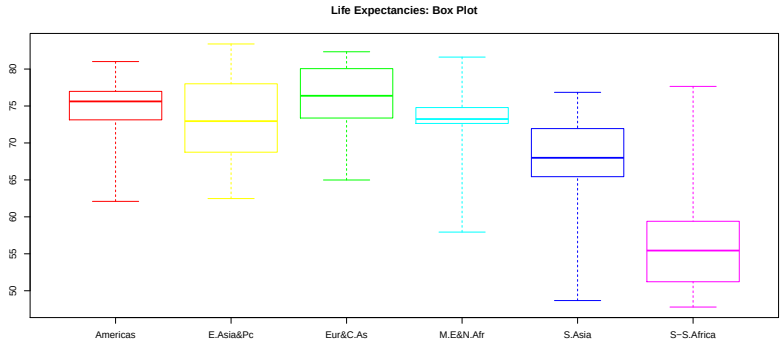
Now we compare 'EAP' with 'MENA' regions, and we want to build simple and modified boxplots in one graph. Note that to plot a modified boxplot we just omit 'range=0' from the arguments. Also to make two charts in one plot we just enter 'par(mfrow=c(1,2))' before the plot functions ('c(1,2)' just indicates that we have two plots in one row).

```
par(mfrow=c(1,2))
boxplot(list(LifeExp[Region=='EAP'],LifeExp[Region=='MENA']),range=0,
   main='Life Expectancies: Box Plot',names=c('E.Asia&Pc','M.E&N.Afr'),
   border=c('blue','violet'))
boxplot(list(LifeExp[Region=='EAP'],LifeExp[Region=='MENA']),
   main='Life Expectancies: Modified Box Plot',names=c('E.Asia&Pc','M.E&N.Afr'),
   border=c('blue','violet'))
```



Finally to plot 'Life Expectancy' boxplots for each 'Region' we use the next commands: ('names' variable stores the names of the 'Region' variable, rainbow(6) produces six colorful colors)

```
names=c('Americas','E.Asia&Pc','Eur&C.As','M.E&N.Afr','S.Asia','S-S.Africa')
boxplot(LifeExp~Region,range=0,main='Life Expectancies: Box Plot',names=names,
   border=rainbow(6))
```



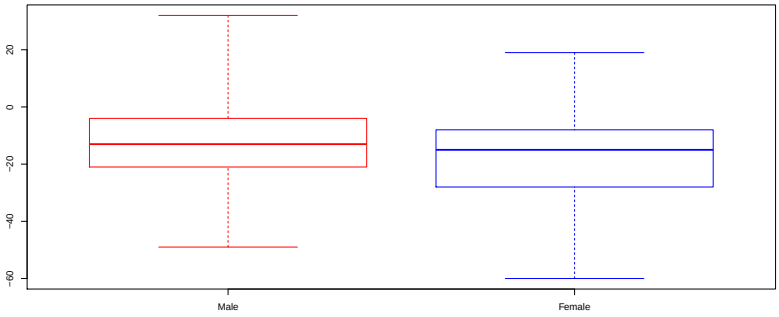
Next lets look at 'Skeleton' data set:

```
Skeleton.data = read.table('SkeletonDataComplete.txt',header=TRUE)
head(Skeleton.data)
attach(Skeleton.data)
```

```
  Sex      BMIcat BMIquant Age DGestimate DGerror SBestimate SBerror
1   2 underweight    15.66  78         44     -34         60     -18
2   1      normal    23.03  44         32     -12         35      -9
3   1  overweight    27.92  72         32     -40         61     -11
4   1  overweight    27.83  59         44     -15         61       2
5   1      normal    21.41  60         32     -28         46     -14
6   1 underweight    13.65  34         25      -9         35       1
```

To get relationship between 'DGerror' variable and 'Sex', we make usual boxplots as described above:

```
boxplot(DGerror~Sex,range=0,names=c('Male','Female'),
  border=c('red','blue'))
```



Next we are interested in how 'DGerror' varies with 'BMIcat'. First we make a table of counts of 'BMIcat':

```
table(BMIcat)
```

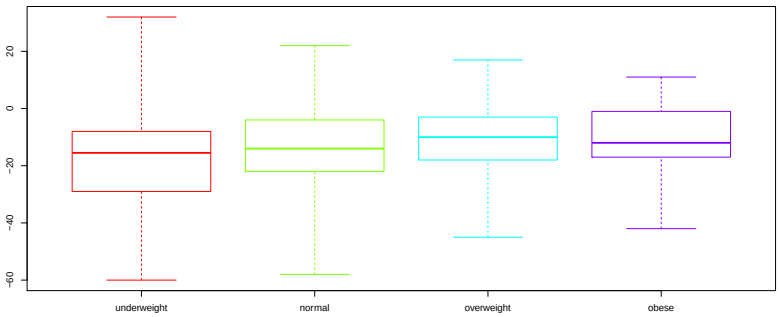
```
BMIcat
     normal       obese  overweight underweight
        225          20          81          74
```

We note that the order is not consistent; we want it to be 'underweight', 'normal', 'overweight' and 'obese'. If we make boxplots of 'DGerror' versus these categories then the order will be also wrong. Hence we modify 'BMIcat' using 'factor' function as follows:

```
BMIcat=factor(BMIcat,c('underweight','normal','overweight','obese'))
```

Now 'BMIcat' is in a right order and we can get the boxplots (it is not necessary to include 'names' argument here since R will get names from factors):

```
boxplot(DGerror~BMIcat,range=0,border=rainbow(4))
```



Relationship between two categorical variables

In this section we show how to work with two categorical variables. We start with 'Skeleton' data; in the last section we have already created variables and we will continue to work with them. The goal is to investigate the relationship between 'BMIcat' and 'Sex' variables (both of them are categorical). First we create a table of counts 'BMI.sex.table' and give names for the 'Sex' factors (since originally they are '1' and '2'):

```
BMI.sex.table=table(BMIcat,Sex)
colnames(BMI.sex.table)=c('Male','Female')
BMI.sex.table
```

```
             Sex
BMIcat        Male Female
  underweight   46     28
  normal       166     59
  overweight    59     22
  obese         10     10
```

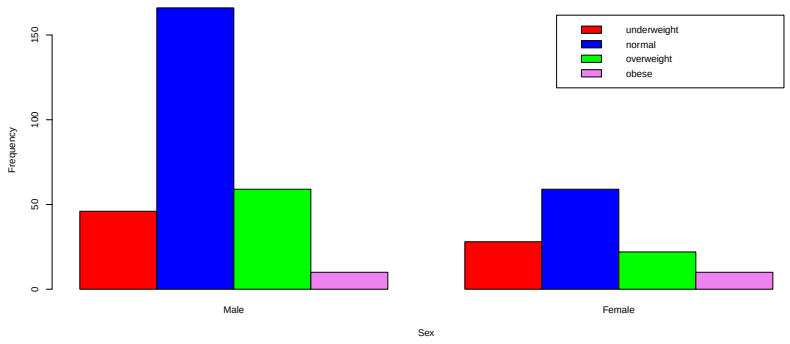
Note that if we just type the object's name (for example BMI.sex.table) then R prints the value of this object. If we want to get joint distribution then we must divide this whole table by total number of observations which is 400 in this case:

```
BMI.sex.table/400
```

```
             Sex
BMIcat          Male Female
  underweight 0.1150 0.0700
  normal      0.4150 0.1475
  overweight  0.1475 0.0550
  obese       0.0250 0.0250
```

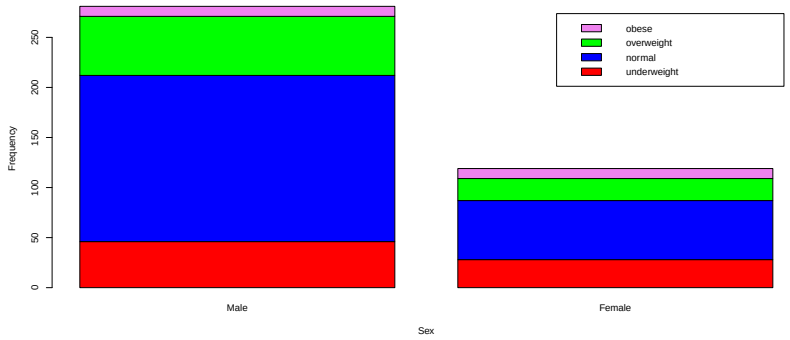
To make a visual representation of the relationship we can use a bar-plot of counts for each 'BMIcat' and 'Sex' categories, using the next command:

```
barplot(BMI.sex.table,beside=TRUE,col=c('red','blue','green','violet'),
  legend=TRUE, ylab='Frequency',xlab='Sex')
```



Note: 'beside=TRUE' specifies that bars are not stacked and 'legend=TRUE' creates a legend at the top right corner. If we want stacked bars then just omit 'beside=TRUE':

```
barplot(BMI.sex.table,col=c('red','blue','green','violet'),legend=TRUE,
   ylab='Frequency',xlab='Sex')
```



These were plots of counts, but next we need to plot conditional distributions of 'BMIcat' given 'Sex' factors. We can do that by dividing the above table by marginal count for 'Sex', or we can

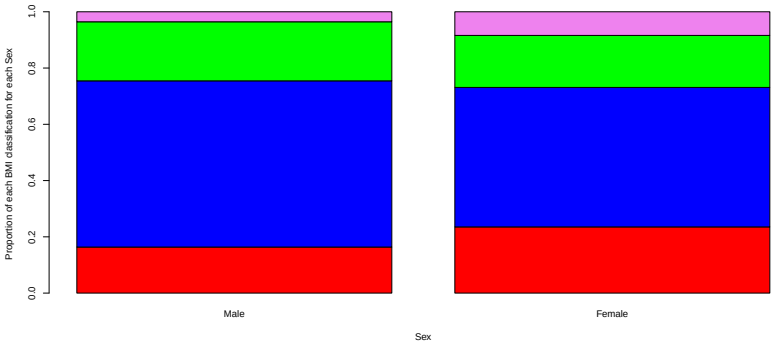
use 'prop.table' function. We call a new table 'BMI.sex.cond', and note that we use '2' in the 'prop.table' function because we need conditional distribution given second variable which is 'Sex'.

```
BMI.sex.cond=prop.table(BMI.sex.table,2)
BMI.sex.cond
```

```
             Sex
BMIcat              Male     Female
  underweight 0.16370107 0.23529412
  normal      0.59074733 0.49579832
  overweight  0.20996441 0.18487395
  obese       0.03558719 0.08403361
```

We see that if we add up each column we get exactly one. Finally we plot a stacked bar-plot for the conditional distribution:

```
barplot(BMI.sex.cond,col=c('red','blue','green','violet'),
  ylab='Proportion of each BMI classification for each Sex',xlab='Sex')
```



Relationship between two quantitative variables

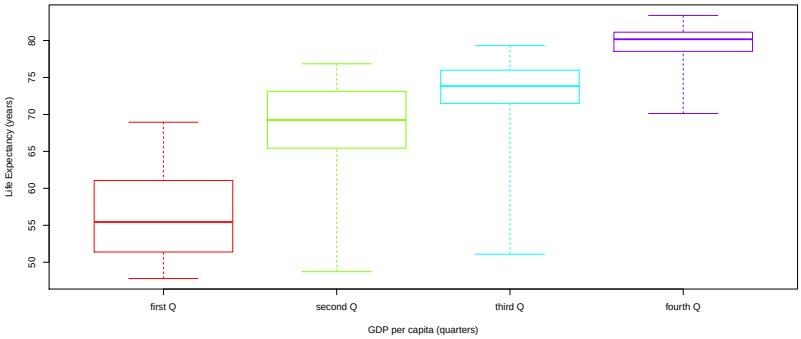
Lets start with the 'Life Expectancy' data set. The goal is to investigate the relationship between 'GDP' and 'LifeExp' (both of them are quantitative). First we get basic statistics of 'GDP' variable:

```
summary(GDP)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   425.4  2072.0  6970.0 12330.0 16860.0 93820.0      50
```

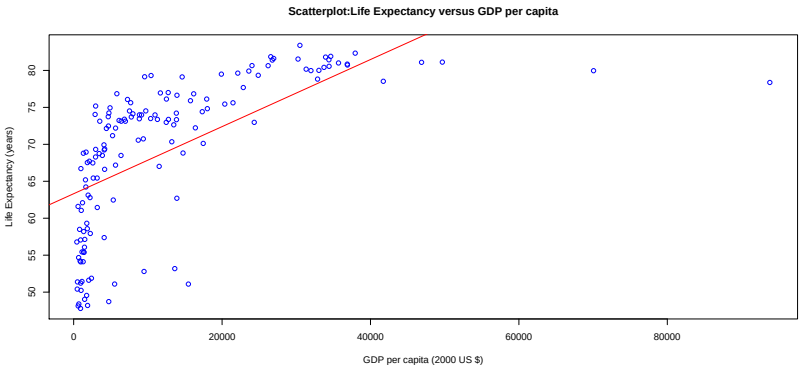
Note that we have 50 'NA's' which means that for 50 observations there are no data about GDP. We can get boxplots of 'LifeExp' for four quarters of 'GDP' as we did in the first section of this document:

```
boxplot(list(LifeExp[GDP<=2072],LifeExp[(2072<GDP)&(GDP<=6970)],
  LifeExp[(6970<GDP)&(GDP<=16860)],LifeExp[16860<GDP]), range=0,
  names=c('first Q','second Q','third Q','fourth Q'), ylab='Life Expectancy
  (years)',xlab='GDP per capita (quarters)',border=rainbow(4))
```



As explained before 'list' function joins several variables together and '&' signs means 'and', so that '$LifeExp[(2072 < GDP)\&(GDP <= 6970)]$' takes into account those values of 'Life Expectancy' for which 'GDP' is greater than 2072 but less than or equal to 6970 (second quarter). However we can also make a simple scatterplot of 'LifeExp' versus 'GDP' using 'plot' function. We can also add a best fit line to the scatterplot using 'abline' function:

```
plot(GDP,LifeExp,col='blue',ylab='Life Expectancy (years)',xlab='GDP per capita
  (2000 US $)', main='Scatterplot:Life Expectancy versus GDP per capita')
abline(lm(LifeExp~GDP),col='red')
```



Here 'lm(LifeExp GDP)' calculates parameters of the line of best fit (much more about this object will be discussed in the regression module). To get correlation statistic between these two variables we use 'cor' function, since there are some missed values in 'GDP' variable we should put 'use="pairwise.complete.obs"' in the argument.

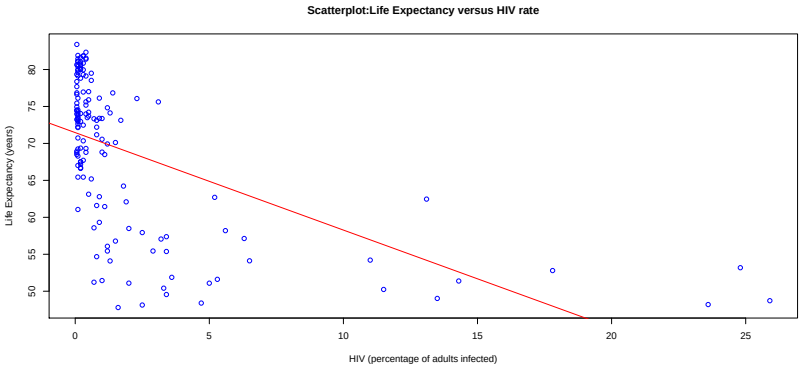
```
cor(LifeExp,GDP, use="pairwise.complete.obs")
```

```
[1] 0.6350906
```

Note: the correlation is positive and therefore the line of best fit increases. Similarly we make a scatterplot and find correlation for 'LifeExp' versus 'HIV':

```
plot(HIV,LifeExp,col='blue',ylab='Life Expectancy (years)',xlab='HIV
  (percentage of adults infected)',main='Scatterplot:Life Expectancy versus HIV rate')
abline(lm(LifeExp~HIV),col='red')
cor(LifeExp,HIV, use="pairwise.complete.obs")
```

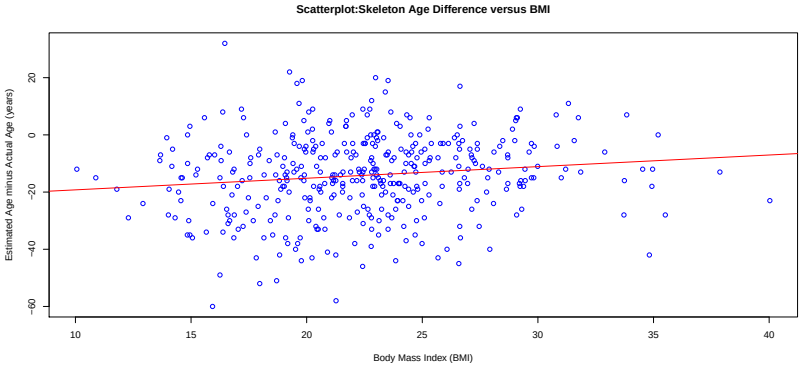
```
[1] -0.5660262
```



In this case we observe negative association. Finally we move to 'Skeleton' data set, once again we make a scatterplot (with the line of best fit) and find correlation for 'DGerror' versus 'BMIquant' variable:

```
plot(BMIquant,DGerror,col='blue',ylab='Estimated Age minus Actual Age (years)',
  xlab='Body Mass Index (BMI)',main='Scatterplot:Skeleton Age Difference versus BMI')
abline(lm(DGerror~BMIquant),col='red')
cor(DGerror,BMIquant)
```

```
[1] 0.1364924
```



Note that there are not missed values in 'DGerror' and 'BMIquant' variables and hence there is no need to put 'use="pairwise.complete.obs"' in the argument of 'cor' function.

Summary of R Functions

We give a short summary of all new and/or important R functions [and arguments] that we used in this Module:

Plots

```
abline() [col]
barplot() [beside,col,legend,xlab,ylab,main]
boxplot() [range,names,border,xlab,ylab,main]
par(mfrow=c( , ))
```

Tables

```
factor()
prop.table()
table()
```

Miscellaneous

```
cor() [use="pairwise.complete.obs"]
```